Structure-Preserving Parametric Deformation of Legacy Geometry

Daniel C. Berkenstock[*]

Stanford University, Stanford, CA, 94305, USA

Michael J. Aftosmis[†]

NASA Ames Research Center, Moffett Field, CA, 94305, USA

We develop a CAD-free geometry engine for shape optimization of complex, three-dimensional geometries represented by surface discretizations. This method permits shape optimization of geometries for which no a-priori parametric CAD representation exists. The geometry engine consists of three parts: geometric preprocessing to improve discretization quality, parametric definition of deformation modes, and a physics-based deformation engine. The deformation engine proceeds by analogy to the elastic behavior of thin plates and shells, mimicking the natural behavior of engineering materials under applied loads and strains. The parametric definition of the deformation modes is performed automatically by decomposing the initial geometry into patches upon which loads, strains, and constraints may be applied. These parameters are created hierarchically in order to allow both flexibility and efficiency during shape optimization. We provide example deformations using this geometry engine in conjunction with surfaces represented by tens and hundreds of thousands of triangles and timings on the order of several minutes using commodity hardware.

I. Introduction

A number of optimization techniques have been successfully employed to improve vehicle performance during the engineering design process. In order to reduce the amount of manual effort and expert knowledge required during problem formulation, we have developed a CAD-free geometry engine for shape optimization that combines the efficiency of parametric CAD with the flexibility of discrete surface representation.

Geometry is often defined through parameters that specify the size, shape, and location of basic building blocks. This is the case when optimization is coupled with parametric Computer-Aided Design (CAD) packages, where parameters such as sweep, wing span, or fuselage diameter offer natural control over complex geometries. This approach to geometry definition offers a ready set of design variables for use during the shape optimization process.

While this CAD-based approach is attractive, there are several situations in which a CAD-free option can ease problem setup and provide greater flexibility during design. First, parametric CAD models are not always easily obtainable for geometries of interest. Often, triangulated surface meshes are the only representation available for so-called *legacy geometries*, shapes derived during previous analysis or design studies. In order to compare such studies to contemporary design tools, a parametric model must be re-created in a current CAD package. Since this may require a large amount of manual effort, a geometry engine that operates on pre-existing surface triangulations alone becomes an attractive alternative. Additionally, the use of triangulated surfaces to represent geometry removes difficulties associated with conversion between different CAD packages and versions. It also eliminates compatibility issues associated with CAD packages that run on operating systems outside the high performance computing environment.

Furthermore, the flexibility of CAD-based design tools to identify a global optimum is dependent upon the parametric definition of the initial geometry. Creating a good set of parameters for complicated geometries

[*]Graduate Student, Department of Aeronautics and Astronautics, AIAA Student Member
[†]Research Scientist, AIAA Senior Member

can require expert knowledge during model creation. There is also a tradeoff between the size of the parameter set, more parameters offer greater flexibility in shape control, and the computational cost of the overall optimization process, since the number of steps to the optimal solution generally increases with number of design variables. This adds a further level of "up front" expert input to parameterize an initial geometry in a manner that offers both flexibility and efficiency in the design process.

In order to alleviate these requirements for manual labor and expert knowledge we investigate the utility of a CAD-free geometry engine that operates on surface triangulations and incorporates an automated approach to parameter set identification. There are three primary elements that we require to define this engine: mesh preprocessing, parameter set identification, and a shape deformation operator. Mesh pre-processing techniques improve triangulation quality in order to reduce discretization errors during subsequent numerical analysis. The parameter set offers a ready pool of design variables. The deformation operator then uses the base geometry and design variables to create modified geometries for analysis during optimization.

A number of preprocessing techniques exist to improve the triangulation quality in surface discretizations that may contain defects detrimental to numerical analysis. These are primarily meant to improve non-topological properties associated with the quality of the local surface tessellation such as high aspect ratio triangles, degenerate edges, and high valence vertices.[1,2] These methods have been extended to identify and preserve geometric features such as creases and corners.

Several deformation operators and parameter set identification techniques have been investigated for use in optimization of geometries represented by triangulated surface meshes. Hicks-Henne bump functions[3] were used by LeGresley and Alonso[4] in airfoil design using Proper Orthogonal Decomposition. Samareh[5] employed freeform deformation based on trivariate tensor-product volume splines for optimization of a variety of three-dimensional aerodynamic shapes and has provided an excellent survey of shape parameterization techniques.[6] Desideri and Janka[7] extended the freeform deformation approach to a hierarchical parameter set and applied it to a number of three dimensional design problems for wings and complete aircraft. Radial basis functions provide a similar method for modifying discretized surfaces embedded in volumetric spaces and have been employed by Jakobsson and Amoignon[8] in the optimization of wings. Fudge, Zingg, and Haimes[9] developed a CAD-free system in which B-spline surface patches were fit to the initial geometry and the control net of these patches formed the parameterization. Recently, multiresolution decomposition techniques, with subdivision surface modelers,[10] have been applied by Dube et al.[11] to turbine design.

These methods represent two classes of deformation operators. The first class of operators encode *complete shape.* Operators from this class provide the basis for CAD systems in which constructive solid modelers use a recipe of primitive operations to *construct* complete geometries. Conversely, a second class of operators encode *shape modification.* In this case, the recipe of primitive operations does not contain the full set of information required to construct a complete geometry from scratch, only the information required to *deform* an initial geometry. In either case, adjusting parameters in the recipe generates various instances of the basic geometry and thus the parameter set defines a family of many shapes derived from a common parent.

There are several important distinctions between these two classes of deformation operators. When considering scalability and problem size, parameter sets that encode the complete shape grow in size as the complexity of the geometry increases. This can have a large impact on the efficiency of shape optimization problems by expanding the size of the design space, thus increasing overall computational cost, and adding difficulty in constraint specification. Conversely, if the parameter set only encodes deformations to an initial geometry, the dimension of the design space is set by the scope of allowable deformations.

Many of these deformation-encoding methods suffer from a potential drawback. Optimization of a baseline geometry is often an exercise in improving existing features, for example, modifications to a wing such as change in sweep, span, or airfoil profile. Therefore, a good deformation operator would allow robust control over existing features using as few parameters as possible. This would imply that the parameter set is closely linked to the structural features of the initial geometry. This goal, while intuitive, is difficult to accomplish with many of the deformation-encoding techniques mentioned above because they do not preserve important structural information, i.e. they are not *structure-preserving.* For instance, bump functions add deformations without regard to topological features such as crease lines and therefore require complex constraint specifications for practical applications. Volumetric techniques, in which a surface is embedded in a deformed volume of space, are reliant on the definition of that volume, which does not necessarily relate to structural features of the original geometry.

One interesting approach to shape deformation involves the approximation of an initial surface by an elastic engineering material that responds to the application of loads and strains. There are several advantages to such an approach. First, regions of the initial shape can simply be excluded from the deformation process. This makes geometry constraint specification trivial during the shape optimization process. Second, the loads and strains are applied to patches of the initial geometry. These patches may be automatically created by using curvature or objective function sensitivity information to provide efficient, intuitive control over prominent features of the initial geometry such as wings, tails, etc. Further, these regions may be defined hierarchically, allowing for multi-fidelity control over the initial shape in response to factors such as desired objective function tolerance and/or overall computational resource availability.

We offer several contributions in this paper. First we propose a modified form of the PriMo method of Botsch et al., a nonlinear approximation to the elastic behavior of thin shells and plates, as a deformation operator for CAD-free shape optimization. We then outline an approach for automatically determining a multi-fidelity parameter set for the deformation of complex, three-dimensional geometries using this operator. We also combine these two techniques with preprocessing methods into a geometry engine for use with shape optimization design tools.

The remainder of the paper is organized as follows. Section II describes the PriMo deformation operator. Section III introduces an automated approach to hierarchical parameter identification. Section IV links this parameter set to design variables for use in shape optimization. Section V discusses our approach to geometric preprocessing using feature sensitive remeshing. Finally, Section VI provides several examples demonstrating the geometry engine.

II. Structure-Preserving Shape Deformation Using PriMo

Unstructured surface triangulations have increasingly become the method of choice for complex shape representation in the computer graphics and digital content creation industry.[1] The success of this approach has led that community to heavily investigate methods for high quality deformation of geometries represented in this fashion.

One such class of deformation techniques is obtained by modeling an initial surface as an elastic structure that responds to applied loads and strains. This is accomplished by approximating the surface as a set of thin shells and plates and then applying simplified energy methods to calculate structural deformations. These deformations represent an updated minimum energy configuration in response to applied loads and strains. Thus, this class of methods is referred to as generating a *minimum variational energy surface*.

There are two primary considerations in developing these methods: the fidelity of the elastic approximation to a real structure and the approach to discretization for use with surface triangulations. The equations chosen to represent elastic behavior may range from the full nonlinear finite element method down to highly linearized, simplistic approximations. The discretization approach may be based on a discrete triangulation or fitted analytic surfaces. Clearly, the choice on both fronts has a strong impact on both realism and computational cost during deformation.

The PriMo algorithm of Botsch et al.[13] represents a good tradeoff between these competing measures for aerodynamic shape optimization. This method models a surface as a thin layer of volumetric prisms coupled by a nonlinear elastic energy. The nonlinear energy is calculated by integration of infinitesimal, stretched elastic fibers between prism faces. This integration improves robustness with respect to triangulation quality and the use of volumetric prisms allows minimization of stretching and bending energies, providing a realistic model of the behavior of actual physical structures.

II.A. PriMo Overview

Figure 1 shows the basic steps used in the PriMo algorithm to deform an initial surface, $S = \{a, b, c, d, e, f\}$, to a modified surface, $S' = \{a', b', c', d', e', f'\}$. First, a layer of volumetric rigid prisms is extruded along the vertex normals of an initial polygonal mesh. These prisms are coupled through a nonlinear energy calculated by an integration over abutting prism faces. Note that this means the initial surface represents a minimum energy configuration, since all abutting prism faces are coincident immediately after extrusion along vertex normals. Second, the system is forced from its initial equilibrium by applying loads and/or strains. The motion of individual prisms may also be constrained in order to anchor regions of the initial geometry. Third, a new minimum energy configuration is found by transforming unconstrained prisms in order to minimize the elastic energies introduced by the applied loads and strains. Finally, a deformed surface is constructed by translating unconstrained vertices in the original mesh according to the updated locations of their incident

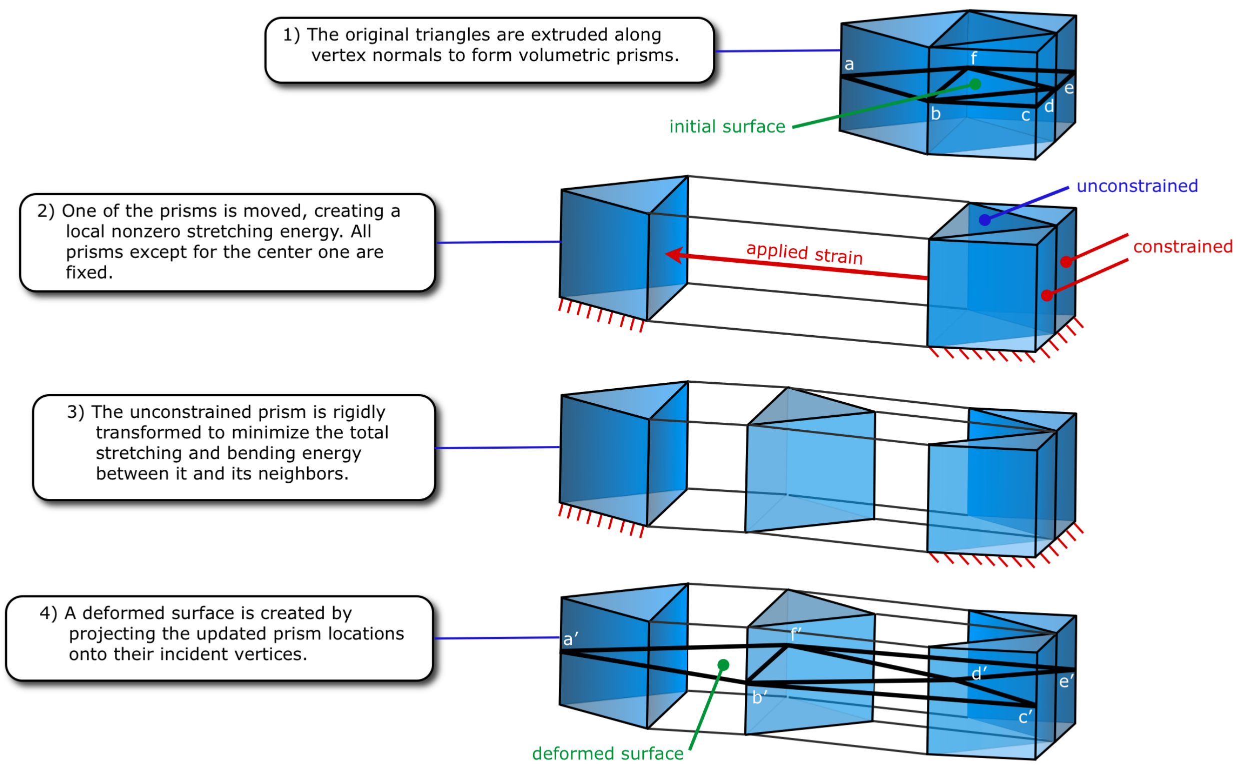
1) The original triangles are extruded along vertex normals to form volumetric prisms.
a
f
e
b
c
d
initial surface
2) One of the prisms is moved, creating a local nonzero stretching energy. All prisms except for the center one are fixed.
unconstrained
applied strain
constrained
3) The unconstrained prism is rigidly transformed to minimize the total stretching and bending energy between it and its neighbors.
4) A deformed surface is created by projecting the updated prism locations onto their incident vertices.
a′
f′
d′
e′
b′
c′
deformed surface

Figure 1. The basic PriMo algorithm consists of four steps: 1) prism initialization, 2) setting boundary conditions to drive deformation, 3) solving for updated prism locations and orientations, and 4) projecting the modified prisms back to a deformed surface.

prisms.

The use of volumetric prisms is integral to the PriMo algorithm in that the prisms allow the calculation of elastic energy over abutting faces to include moment information. This leads to deformations that more closely resemble the behavior of stiff elastic materials. This is because minimizing stretching alone leads to a minimum area surface, while minimizing bending leads to a minimum curvature surface. Minimum curvature surfaces correspond more closely to the natural deformation of structures with stiffness, and therefore the inclusion of both is important to providing a flexible and efficient geometry engine for shape optimization. Furthermore, the height of the prisms is variable, in order to specify stiffness locally, and may be used as an additional parameter in shape control. Although these prisms may overlap in regions of high curvature, the energy calculation is based solely on the integration of a norm describing separation between abutting prism faces. Since this number is always positive, intersections between prisms do not negatively impact the deformation process.

Subsets of the prisms created using several typical legacy geometries are shown in Figure 2. The prisms associated with the wing geometry seem to correspond to a typical volume mesh, whereas the prisms associated with the more complex hypersonic vehicle actually overlap in places. However, since this does not adversely impact the energy calculation, both prism sets are equally acceptable.

II.B. Energy Definitions

During deformation, the PriMo algorithm seeks an updated minimum energy equilibrium in the presence of applied loads and strains. The total energy, E, present in the system is a combination of internal stretching and bending as well as externally applied loads,

$$E = E_{\text{internal}} + E_{\text{external}} \tag{1}$$

The energy associated with internal stretching and bending, which also reflects applied strains,

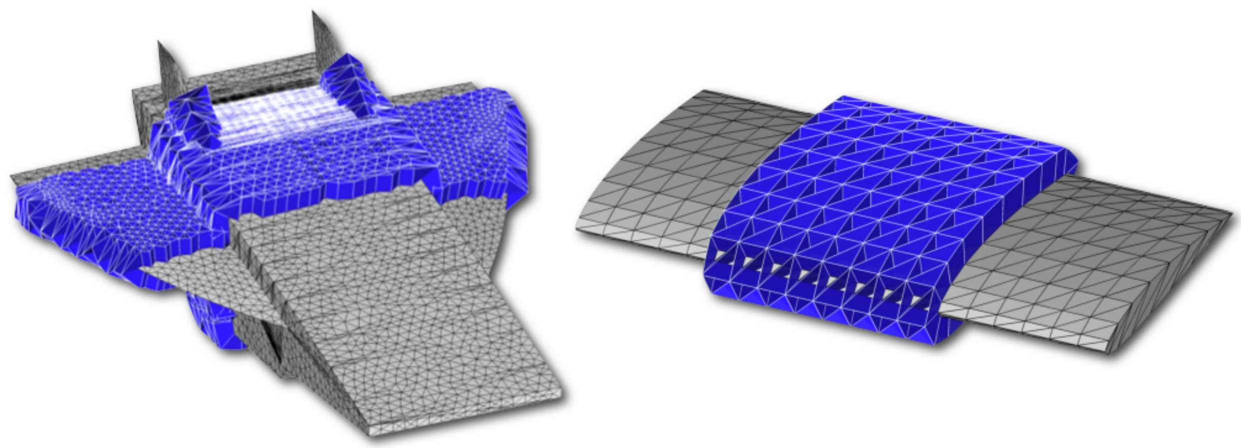

Figure 2. Selected volumetric prisms for hypersonic vehicle and wing (shown extruded outward along vertex normals).

$$E_{\text{internal}} = \sum_{\{i,j\}} w_{ij} E_{ij} \tag{2}$$

is calculated as a summation over all prisms, i and j, with abutting faces. This summation reflects the elastic energy generated by stretching infinitesimal fibers between these faces,

$$E_{ij} = \int_0^1 \int_0^1 ||f^{i\to j}(u,v) - f^{j\to i}(u,v)||^2 \mathrm{d}u\mathrm{d}v \tag{3}$$

and a weighting factor,

$$w_{ij} = \frac{||e_{ij}||^2}{A_i + A_j} \tag{4}$$

comprised of the areas of triangles i and j, A_i and A_j, as well as the length of their shared edge in the original surface triangulation, $||e_{ij}||$.

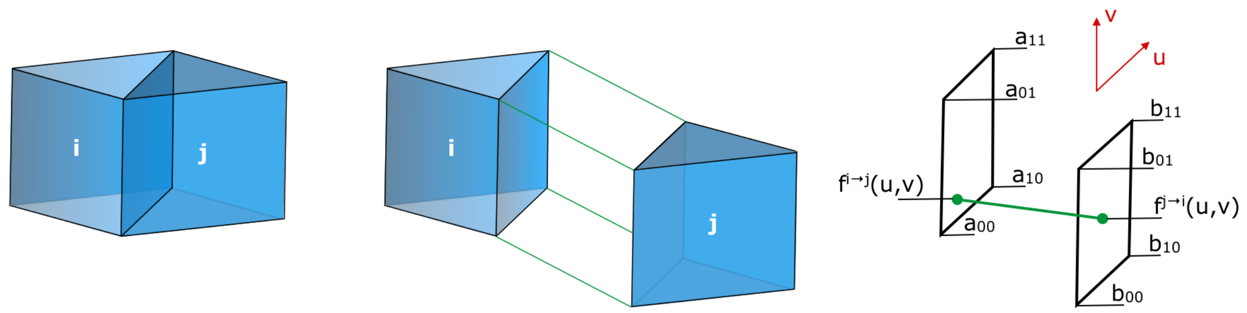

Figure 3. During deformation, two originally abutting prisms, i and j, are separated, creating a local stretching and bending energy. This energy is calculated by the integrated norm between the two abutting prism faces.

Figure 3 illustrates this calculation between two prism faces. The prisms start in an initial minimum energy state with zero stretching and bending energy between the abutting faces. The prisms are then pried apart, creating a local nonzero energy. This energy is quadratic and calculated in closed form, using a bi-linear interpolation over the $u - v$ parameter space of the faces, so that

$$E_{ij} = \int_0^1 \int_0^1 ||f^{i\to j}(u,v) - f^{j\to i}(u,v)||^2 \mathrm{d}u\mathrm{d}v = \frac{1}{9} \sum_{i,j,k,l=0}^{1} a_{ij} \cdot b_{kl} \cdot 2^{(-|i-k|-|j-l|)} \tag{5}$$

Here, $f^{i\to j}(u,v)$ is a point on the face of prism i that abuts prism j. This face is parameterized by $u \in [0,1]$, $v \in [0,1]$, and $f^{j\to i}(u,v)$ is defined similary. $\{a_{00}, a_{10}, a_{01}, a_{11}\}$ and $\{b_{00}, b_{10}, b_{01}, b_{11}\}$ represent the corners of the two prism faces so that, for example, the bi-linear interpolation of the x coordinate function over the face of prism i abutting prism j is,

$$f_x^{i\to j}(u,v) = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} a_{00_x} & a_{01_x} \\ a_{10_x} & a_{11_x} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix} \tag{6}$$

The second term of the total energy, E, from Equation 1,

$$E_{\text{external}} = \sum_{k\in\mathcal{V}} E_k \tag{7}$$

represents a summation of the energies due to externally applied loads on a subset $\mathcal{V}$ of vertices in the initial triangulated surface. The energy at each vertex with an externally applied load is calculated as the product of a spring constant, λ_k, and squared distance between a field point, v_{t_k}, and current vertex location, v_k,

$$E_k = \lambda_k ||v_{t_k} - v_k||^2 \tag{8}$$

This field point,

$$v_{t_k} = v_{k_i} + \frac{F_k}{2\lambda_k} \tag{9}$$

is calculated at the beginning of the deformation process as a function of the applied load, F_k, and initial vertex location, v_{k_i}, of vertex k.

II.C. Solution Technique

Once strains and loads have been applied to the initial prism configuration, solving for a modified surface becomes a geometric optimization problem where we seek to minimize the sum of internal and external elastic energies,

$$\min_{R_i,t_i} E_{\text{internal}} + E_{\text{external}} \tag{10}$$

through rigid body rotations, R_i, and translations, t_i, of all unconstrained prisms, i. Botsch et al. proposed to approximate these rigid motions to first order as,

$$R_i(\cdot) + t_i \approx (\cdot) + \omega_i \times (\cdot) + v_i := A_i \tag{11}$$

where ω_i represents a rotational velocity vector and v_i a translational velocity vector, and solve the quadratic minimization problem,

$$\min_{\{v_i,\omega_i\}} \left(\sum_{\{i,j\}} \int_0^1 \int_0^1 ||A_i(f^{i\to j}(u,v)) - A_j(f^{j\to i}(u,v))||^2 \mathrm{d}u\mathrm{d}v + \sum_{i\in\mathcal{V}} \lambda_i ||v_t(i) - A_i(v(i))||^2 \right) \tag{12}$$

The solution is obtained by setting the partial derivatives, with respect to the translational and rotational velocities of each prism, of this quadratic form to zero. This results in a linear system for an affine transformation of each unconstrained prism. The ensuing equations represent a sparse, symmetric positive definite block system with $6n$ equations for a surface tessellation comprised of n triangles. The system is generally quite sparse as block entries occur only between neighboring triangles and in the local neighborhood of a vertex with a target location.

The original PriMo algorithm was intended for use with interactive applications and, thus, was limited to solution methods with timings on the order of several seconds. In order to reduce the solution time for dense triangulations, a mesh sequencing approach was used where the actual linear system was only solved on a coarsened version of the mesh. Several sweeps of a nonlinear relaxation operator were also applied on each finer level of the hierarchy, resulting in an approximation to the overall nonlinear deformation operator.

Since objective function solutions in aerodynamic optimization problems are expensive, we can afford a deformation operator with greater computational cost. Therefore, in Algorithm 1, we propose a modified solution approach with the principle difference that we solve the linear system iteratively on the *original, fine mesh* until the scale factor of the displacement vector between linear solutions drops below a sufficiently small ϵ. These linear updates are solved using the CHOLMOD solver distributed with SuiteSparse.[14,15,16,17,18] While efficient, this direct method still faces scalability issues with densely discretized geometries. In order to mitigate such limitations in the future, we are currently implementing an agglomeration multigrid solver, for the solution of the linear iteration, to reduce memory requirements and allow parallelization.

Algorithm 1 Modified PriMo Algorithm

```
 1: Initialize volumetric prisms
 2: $\nu = 1.$
 3: while $\nu > \epsilon$ do
 4:     Calculate $E_0$ = global energy
 5:     Populate linear $Ax = b$ system
 6:     Solve for $x$
 7:     Apply rigid update $\nu x$
 8:     Calculate $E_f$ = global energy
 9:     while $E_f > E_0$ && $\nu > \epsilon$ do
10:         $\nu = \nu/2$
11:         Apply rigid update $\nu x$
12:         Calculate $E_f$ = global energy
13:     end while
14: end while
15: Project modified prisms to deformed surface
```

Furthermore, the linearized affine transformation associated with each unconstrained prism results in a local shape matching problem to determine a closest rigid transformation. The use of rigid transformations prevents degradation of mesh quality during the transformation process and increases overall robustness of the deformation operator.[13] The rotational and translational velocities are scaled by a factor, ν, during each solution step, to the maximum value that reduces the global energy. Once ν is less than ϵ, the process has converged and unconstrained surface vertices are translated to the mean location of their incident prism faces.

III. Automated Parameter Discovery

We make the observation that optimization of a baseline geometry is generally an exercise in improving existing features. Therefore, an automated parameterization should capture, and allow modification of, initial components and features within the geometry.

When using the PriMo deformation operator, shape modifications are initiated by applying boundary conditions over patches of triangles. The parameters define these boundary conditions which closely links the parametric definition to the underlying patches. Therefore, a method to decompose monolithic triangulations into patches that capture features or components of the geometry offers an intuitive approach for automatically discovering a parameter set.

In order to create these patches we require two components: a method to identify boundaries between patches and a method to automatically generate a hierarchical set of these patches.

III.A. Curvature-Based Patch Boundaries

Regions of high curvature typically define boundaries between intuitive components of a geometry. Therefore, one potential method is a decomposition of monolithic triangulations into patches of similar curvature. However, producing accurate estimates of curvature over poor quality surface triangulations can prove problematic due to discretization of the chosen curvature measure.

For example, one common surface Laplacian, the cotangent formulation of the Laplace-Beltrami operator, evaluates to the mean curvature normal when applied to the coordinate function of a surface,[1]

$$\triangle_S f(v) := \frac{2}{A(v)} \sum_{v_i \in \mathcal{N}_1(v)} (\cot \alpha_i + \cot \beta_i)(f(v_i) - f(v)) \tag{13}$$

where $A(v)$ is the Voronoi area surrounding vertex v, v_i is a vertex in the one-ring of v, $\mathcal{N}_1(v)$, f is the coordinate function, and α and β are the interior angles opposing the edge connecting v and v_i. This formulation suffers from the defect that the cotangent terms become negative in the presence of obtuse triangles, which rapidly degrades the accuracy of such methods in general triangulations.

To mitigate these defects, Jiao et al.[19] proposed a robust method for feature detection and curvature estimation based on the quadric metric tensor, A,

$$A = N^{\mathrm{T}} W N \tag{14}$$

where $N \in \Re^{mx3}$ is a matrix with rows containing the normal vectors of the m triangles neighboring vertex p and W is a diagonal matrix of weights. These weights are taken to be the angle, in radians, in each triangle of the one-ring centered at vertex p.

An eigenanalysis is performed in order to classify each vertex. In general, the quadric metric tensor has three large eigenvalues at a corner, two large eigenvalues at a crease, and one large eigenvalue at a point on a smooth surface. The number of large eigenvalues at a given vertex is determined by a ratio threshold that loosely corresponds to the dihedral angle, τ, across an edge.

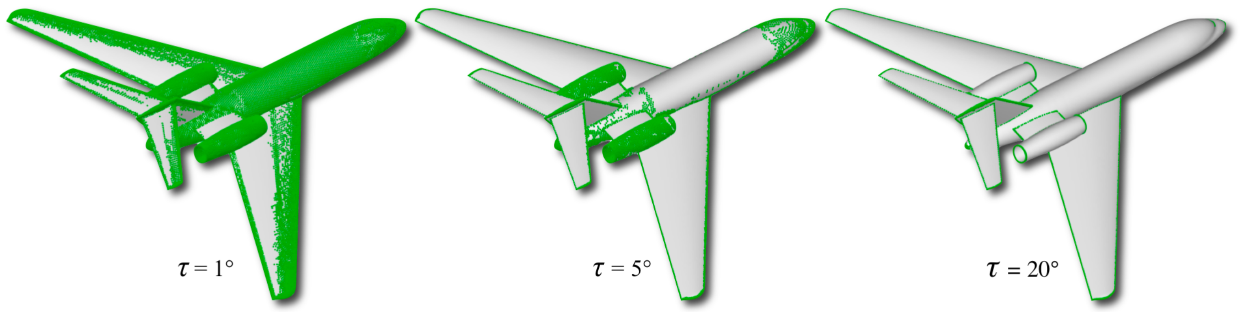

Figure 4. QMT based vertex classification: nonsmooth vertices, τ = 1°, 5°, and 20°.

Figure 4 shows vertices classified as nonsmooth using this approach with increasing angle threshold, τ. After vertex classification, any edge in the triangulation with two nonsmooth vertices and dihedral angle above a minimum value is also classified as nonsmooth.

III.B. Hierarchical Patch Segmentation

Patches are identified by seeding smooth regions and painting outwards to nonsmooth edge boundaries. Once all smooth patches have been identified for a given curvature threshold, the nonsmooth regions are combined into patches in a similar manner. By starting with a large initial threshold and then varying it we are able to automatically develop recursive patches at different scales. Patches with less than a given percentage of the total number of surface triangles are filtered out and combined with larger neighbors of the same curvature classification.

There are several possible approaches to identify patches for decomposition at each level in the hierarchy. One automated approach is to simply split them all. However, this quickly results in a large number of surface patches, which degrades the efficiency of using these patches as handles to the deformation operator. A second automated approach is to decompose the largest, or several largest, patches at each level. Once an objective function has been defined, another automated approach would be to decompose the patches for which the objective function has the the greatest integrated sensitivity. This would allow automatically provide higher fidelity surface control in regions of the geometry with the greatest impact on the final design. A number of interactive approaches are possible as well, such as decomposing the patch at each level containing a specified triangle of the original surface. This is the approach taken in Figure 5 where a triangle on the wing of the initial geometry is selected and the patch containing that triangle is further decomposed at each level in the hierarchy. This approach generates detailed control over the surface of the wing while offering lower fidelity control over regions of the geometry with less influence on an objective function such as lift to drag ratio. Timings provided were performed on a desktop workstation.[a]

[a]3GHz dual quad-core Intel Xeon, single thread

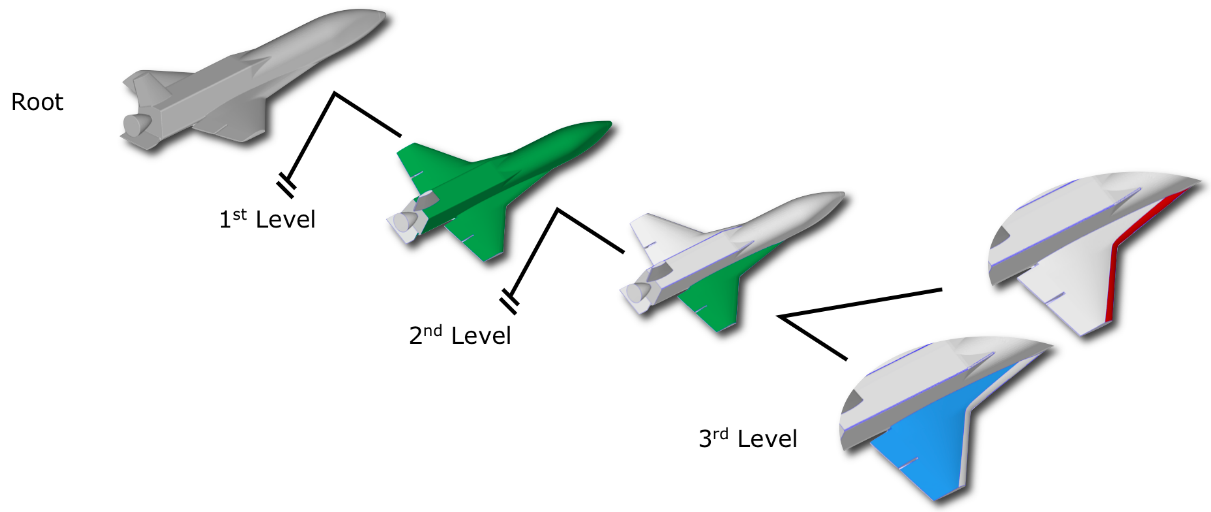

Figure 5. Hierarchical segmentation of hypersonic vehicle comprised of 418k triangles. Decomposition performed on a desktop workstation: level 1 - 9.5 sec, level 2 - 80 sec, level 3 - 11.5 sec.

IV. Shape Deformation Using Design Variables

The deformation and parameter set identification methods described above are intended for integration with design tools that solve a shape optimization problem which consists of determining values of design variables, X, that minimize a given objective function, $\mathcal{J}$,

$$\min_{X} \mathcal{J}(\mathcal{D}(M, X)) \tag{15}$$

where $\mathcal{D}$ is the deformation operator, M is the original, unmodified surface triangulation, and the design variables, X, represent applied loads and strains over the surface patches described in Section III. The full parametric definition of deformation modes consists of applied loads and strains over each individual patch. The set of design variables, X, is a subset of the full parameter set that may be either selected by a designer or generated automatically, e.g. all parameters controlling the shape of a wing or other components for which the objective function has high sensitivity.

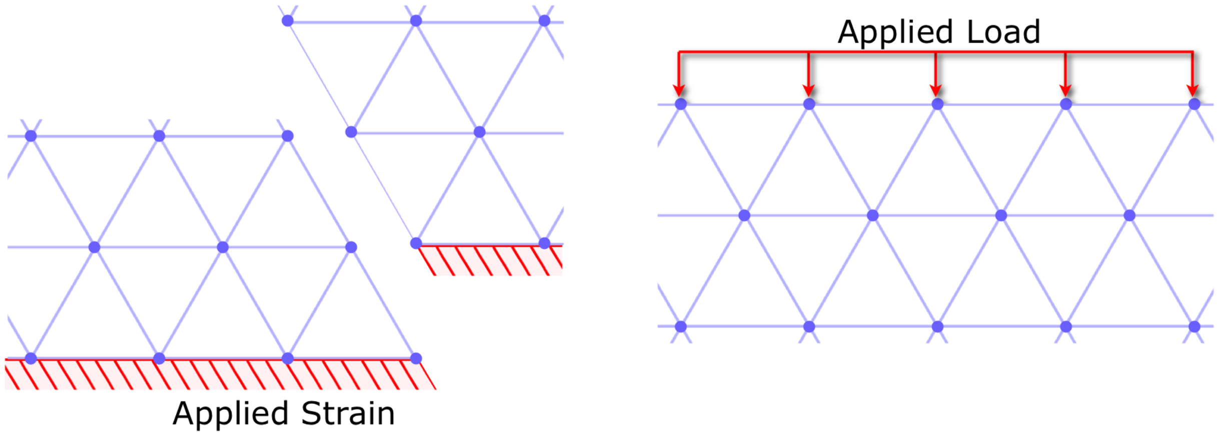

Figure 6. Design variables in the shape optimization process correspond to applied loads and strains.

Figure 6 shows the two types of boundary conditions that may be used to form design variables for the shape optimization process. Applied strains are defined by translating or rotating sets of prisms away from their initial resting condition and then pinning them. Applied loads are defined over sets of vertices within the initial surface discretization.

V. Isotropic Remeshing

In order to deform general surface triangulations of unknown origin and quality, we must mitigate conversion defects and other discretization deficits such as degenerate edges, high valence vertices, and highly obtuse angles. We address these issues by applying a feature preserving, isotropic remeshing algorithm before any other operations in our geometry engine. This algorithm iteratively collapses short edges, splits long edges, smoothes the tangential distribution of vertices, and swaps edges to minimize valence or eliminate obtuse angles. The algorithm is described in detail in the Siggraph 2007 Course Notes,[20] however, we outline it below in Algorithm 2.

Algorithm 2 Isotropic Remeshing

```
1:  INPUT: General three dimensional triangulated surface mesh
2:  CollapseDegenerateEdges()
3:  ClassifyAllVertices()
4:  while iteration $\leq$ maxNumberIterations do
5:      SplitAllEdgesLongerThan( 4/3l )
6:      CollapseAllEdgesShorterThan( 4/5l )
7:      SwapEdgesToReduceMaximumAngle()
8:      TangentialSmooth()
9:  end while
10: OUTPUT: Approximately isotropic mesh retaining original feature edges and corners
```

To preserve features such as creases and corners, we apply the classification algorithm of Jiao et al.[19] before any geometry modifications. During subsequent remeshing operations, crease edges may only be collapsed along a crease edge, crease vertices are only smoothed tangentially along the crease, and crease and corner edges are excluded from all swapping operations.[1]

In addition to preservation of high-curvature features, all geometry operations are checked to determine that the proposed operation does not violate the topology of the surface. Operations are also post-checked by comparison of cosine between pre- and post-normals of all modified triangles. Generally, any operation that would result in a change in normal vector greater than a tolerance on the order of $1 - 2^o$ is rejected.

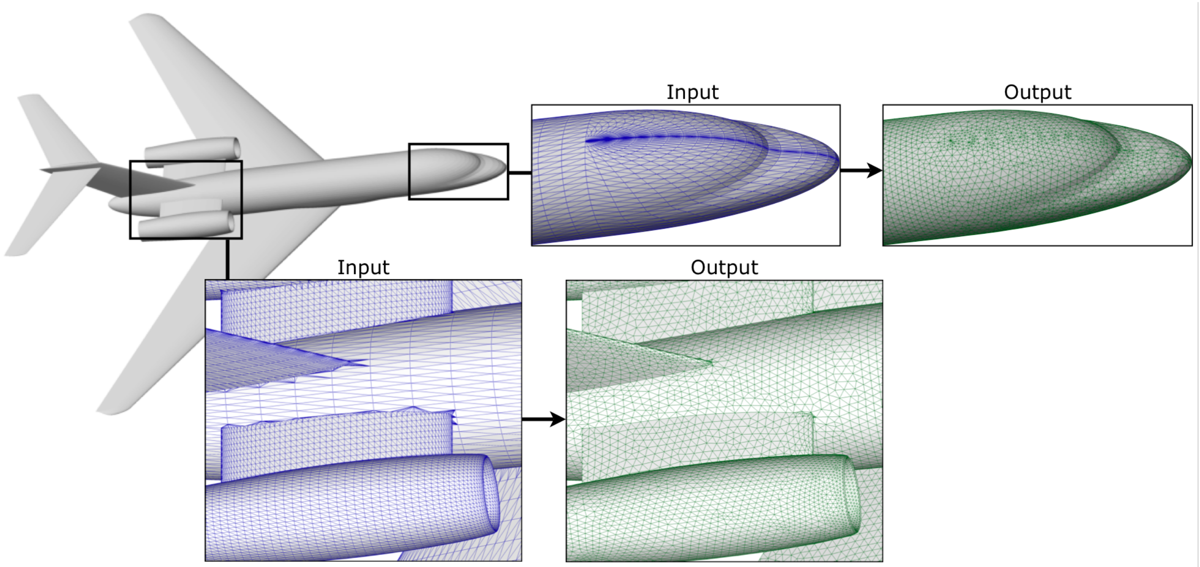

Figure 7. Feature-sensitive, isotropic remeshing of a business jet legacy geometry - performed on a desktop workstation in 18 seconds. Initial surface triangulation (shown in blue): 93k triangles. Nearly isotropic, improved mesh (shown in green): 112k triangles.

Figure 7 shows the resulting surface discretization of this algorithm applied to a business jet legacy geometry. The initial surface tessellation contains a number of undesirable conversion defects, such as high valence vertices and highly obtuse triangles. The remeshing algorithm mitigates these defects while preserving creases, and other topological properties, of the geometry.

VI. Examples

The methods described above have been combined into a geometry deformation engine written in ANSI C. We show several examples below demonstrating the process of parameter set identification and shape deformation.

VI.A. 3D Wing Geometry

The first example legacy geometry is a three dimensional wing surface triangulation containing approximately 22k triangles. The wing was isotropically remeshed, in 8.5 seconds on a desktop workstation, resulting in a surface triangulation also containing approximately 22k triangles. A parameter set was created by defining patches based on 10^o and 1^o thresholds. The components derived from this segmentation, shown in Figure 8, separate the wing root, the upper and lower surfaces, and a number of narrow high curvature patches near the leading edge.

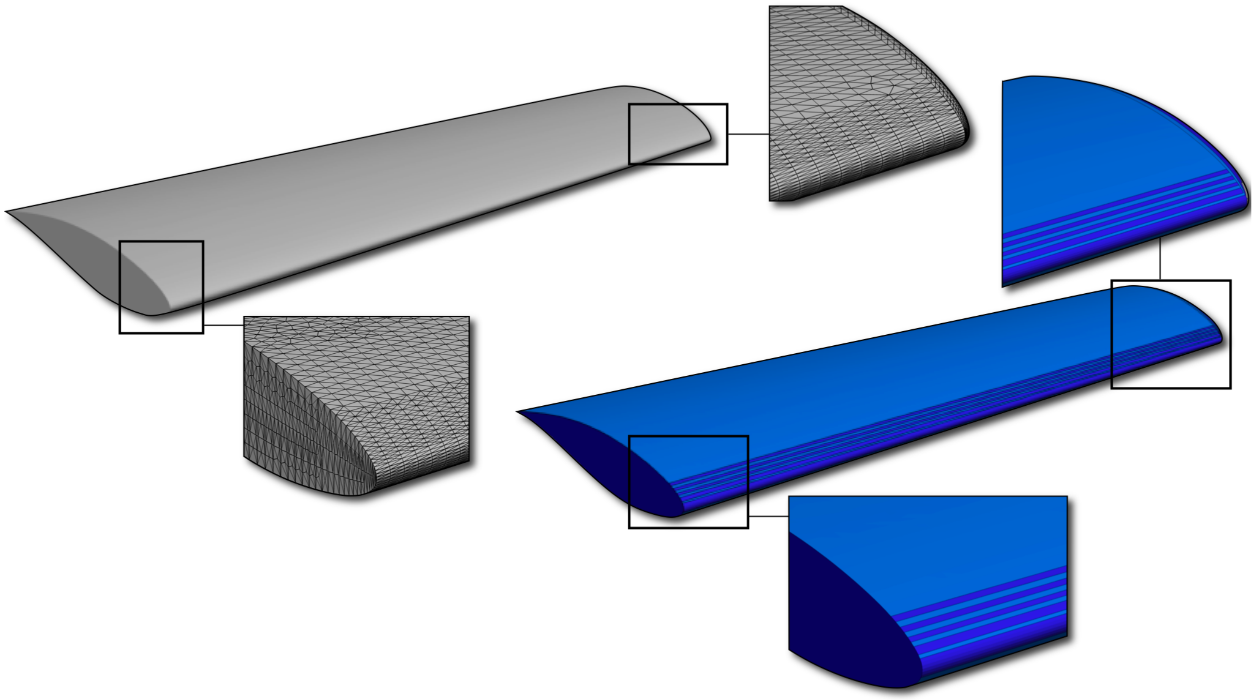

Figure 8. Three dimensional wing and patch decomposition.

Figure 9 shows the deformation result of an applied load to the top wing surface that loosely corresponds to sweep. Similarly, Figure 10 shows the deformation result of an applied load to the top wing surface that loosely corresponds to twist and dihedral. For small deformations, a single linear solve, effected in about 10 seconds on a desktop workstation, suffices. Large scale deformations of a nonlinear nature, require several iterations of the linearized solution, in total approximately 55 seconds on a desktop workstation.

VI.B. Hypersonic Lifting Body

A second example shows the deformation of a pre-existing hypersonic lifting body consisting of approximately 131k triangles. The automatically determined set of parameters, shown in Figure 11, was developed using a recursive approach with $\tau = 10^o$, 5^o, and 2.5^o. The patches obtained correspond to obvious vehicle features,

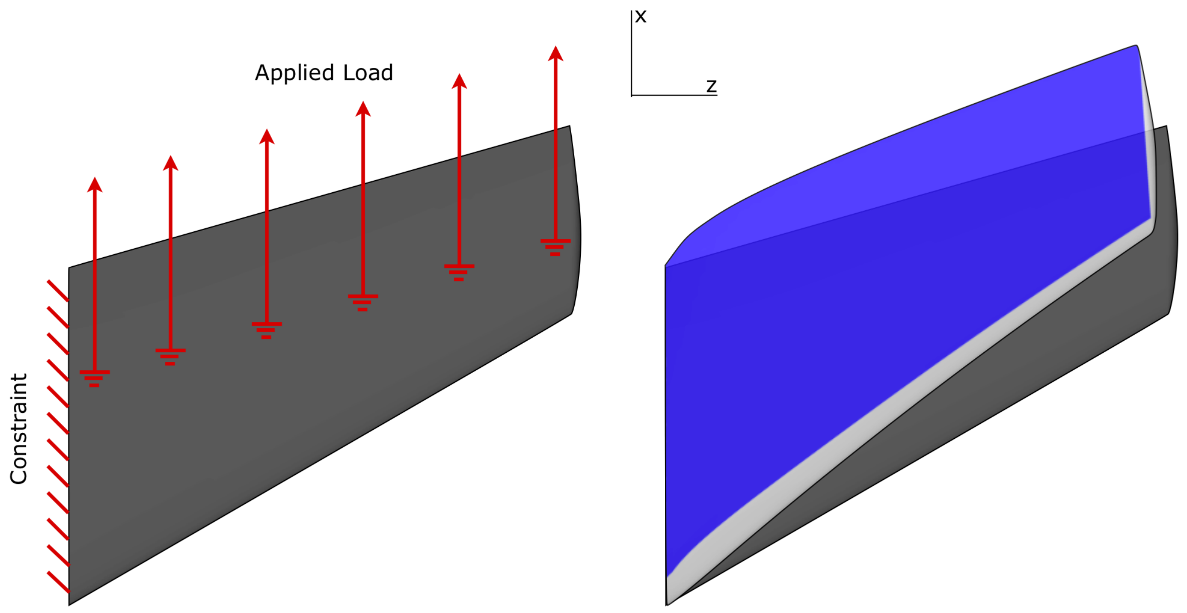

Figure 9. Uniform shear applied in x direction to top of wing: 23.5k triangles. 10 seconds for single linear solve on a desktop workstation, 55 seconds for full nonlinear solve. 3.9M sparse matrix entries with average of 28 entries per row (modified geometry shown in blue).

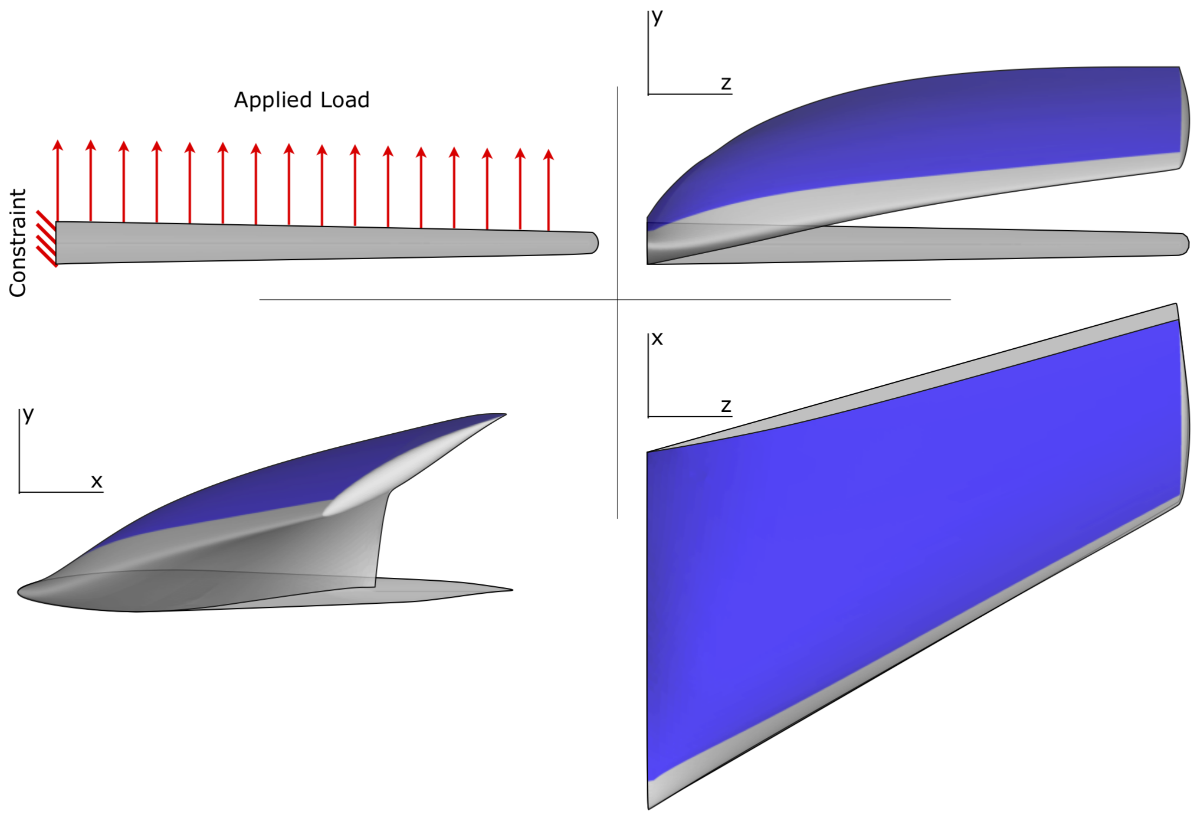

Figure 10. Uniform force applied in y direction to top of wing: 23.5k triangles. 10 seconds for single linear solve on a desktop workstation, 55 seconds for full nonlinear solve. 3.9M sparse matrix elements with average of 28 entries per row (modified geometry shown in blue).

such as cockpit and wingerons, as well as less intuitive segmentations of the main body lifting surface along regions of similar curvature.

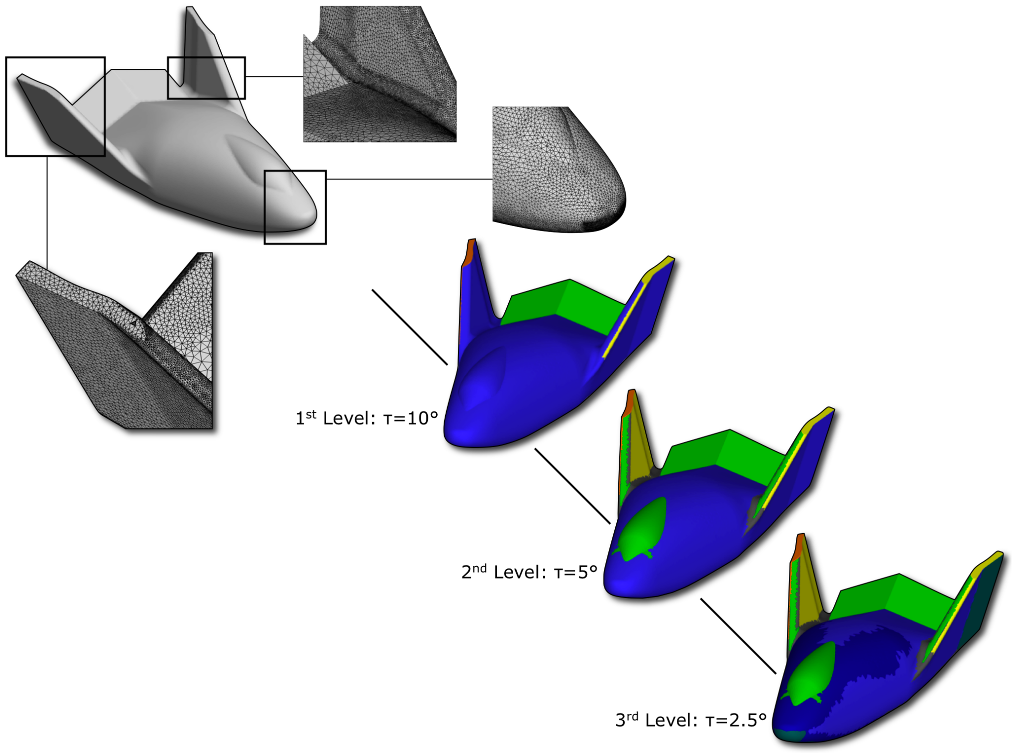

Figure 11. Hypersonic lifting body legacy geometry: 131k triangles. Parameter set identified in 80 seconds on a desktop workstation.

Figure 12, left, shows boundary conditions applied for a representative surface deformation. Regions in the nose and flap sections were constrained, in order to prevent rigid body rotations, and an applied load was defined over the cockpit region.

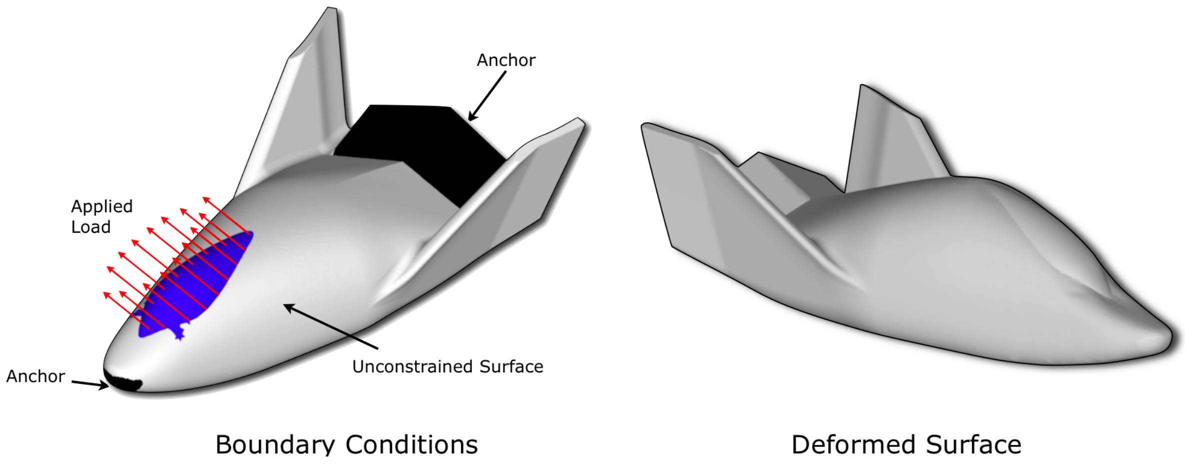

Figure 12. Hypersonic lifting body legacy geometry: 27 seconds for single linear solve, 340 seconds for full nonlinear solve on a desktop workstation. 9.3M sparse matrix elements with average of 12 entries per row.

The full nonlinear PriMo equations were solved iteratively until the 2-norm of the displacement vector between linear solutions was on the order of 10^{-7}. The resulting deformation, Figure 12, right, highlights the large amplitude modifications of legacy surface triangulations possible with the PriMo deformation operator. The volume is markedly increased with a large change in the region near the cockpit and smooth transition near constrained patches.

VI.C. Stanford Bunny

The final demonstration case is a legacy triangulation of the “Stanford Bunny”, a surface comprised of approximately 70k triangles which was obtained from the Stanford University Computer Graphics Laboratory.[21] After closing several holes in the underside of the surface discretization, using MeshLab,[22] and performing isotropic remeshing, using the method of Algorithm 2 presented in Section V, the triangulated surface shown in Figure 13 contains approximately 83k triangles. This triangulation was decomposed into 9 surface patches using a threshold of $\tau = 10^{o}$.

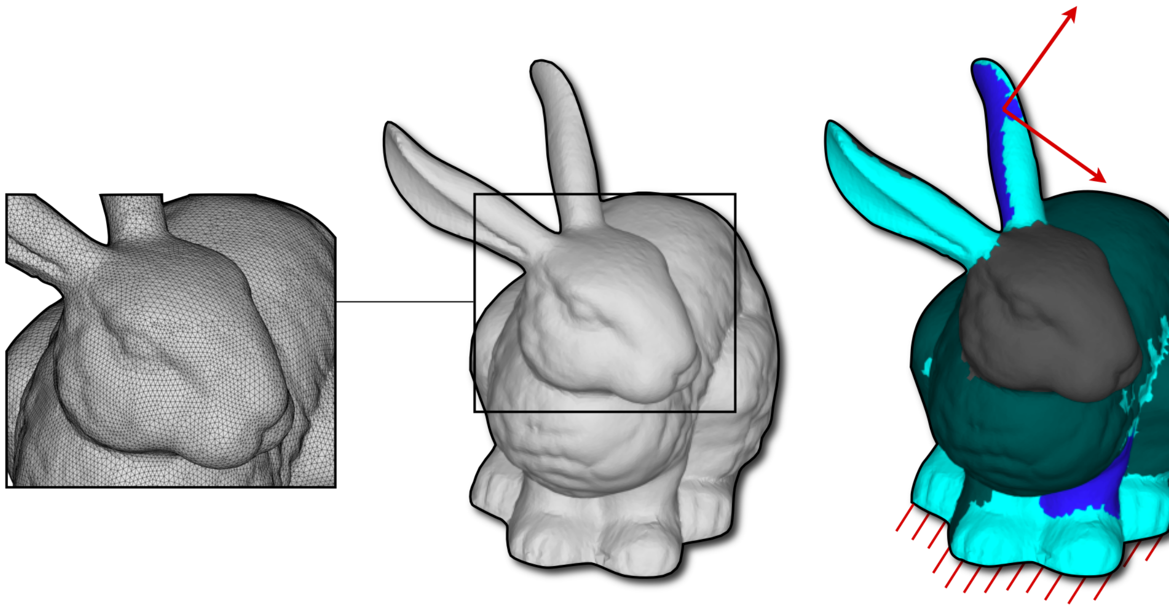

Figure 13. The “Stanford Bunny”: isotropically remeshed and decomposed with $\tau = 10^{o}$, performed in 8.3 seconds on a desktop workstation. Loads applied to ear and base constrained.

The decomposition corresponds loosely to prominent structural features such as the bunny’s hear, ears, body, and base. Pursuing a hierarchical decomposition would also provide higher fidelity control over components such as the eyes, nose, and feet.

The boundary conditions to the PriMo algorithm include a load applied to the bunny’s left ear backwards and to the left, as well as a constraint applied to the base of the bunny. Figure 14 shows how the textured surface in this example provides a good example of the structure-preserving quality of deformations effected through the use of PriMo. Although there is a significant change in the shape and location of the ear due to the applied forces, prominent features, such as eyes, feet, and tail, retain small scale detail similar to that of the initial geometry.

VII. Conclusions and Future Work

We have presented a CAD-free geometry engine for use with complex, three-dimensional triangulated surfaces in support of shape optimization. Our primary contributions have been the use of a structure-preserving deformation operator coupled to an automated method for parameter set identification. This deformation operator leads to intuitive shape modifications by modeling a surface as an elastic engineering material that responds to applied loads and strains. These loads and strains are applied over a hierarchical set of patches that are created automatically by decomposing the initial geometry into regions of similar curvature. We have demonstrated the ability of this process to create intuitive surface deformations on geometries comprised of up to 131k triangles with timings on the order of several minutes using commodity hardware.

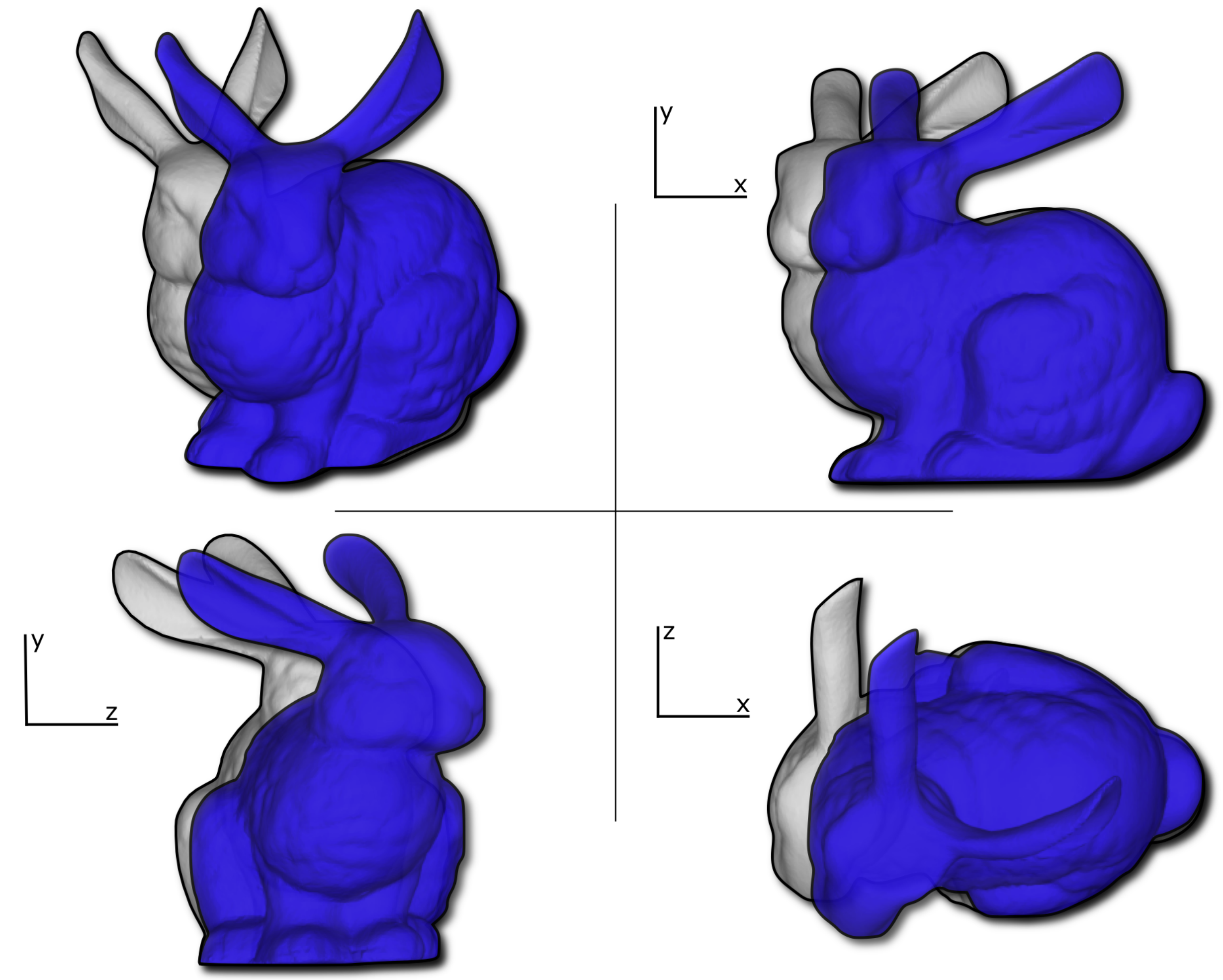

Figure 14. Deformed Stanford Bunny. 31 seconds for single linear solve, 326 seconds for full nonlinear solve on a desktop workstation. 8.3M sparse matrix elements with average of 17 entries per row.

The next goal of this effort is integration with the NASA Cart3D design suite in order to enable shape optimization studies on legacy geometries as well as in situations where linking to CAD or a user-defined geometry generator is not possible. There are several remaining tasks required to fully realize this objective. First, we plan to finalize implementation of an agglomeration multigrid solver for use with PriMo in order to decrease time and memory requirements. Secondly, we plan to improve the parameterization process by linking it to sensitivity information in order to decompose patches at each level in the hierarchy for which the vertex locations have the largest impact on the shape optimization objective function. Finally, we plan to use the recursive nature of the derived parameterization to drive a hierarchical optimization process, starting with parameters representing global changes and iterating to local shape modification.

Acknowledgments

The authors are grateful to Dr. Mario Botsch for sample code and assistance in proper implementation of the PriMo algorithm as well as to Dr. Marian Nemec and Dr. Tom Pulliam for valuable discussions regarding shape optimization problem formulation. Support from NASA Ames, under award NNA07CN71A, is also gratefully acknowledged.

References

[1]Botsch, M., Pauly, M., Kobbelt, L., Alliez ,P., Levy, B., Bischoff, S., and Rössl, C., "Coursenotes: Geometric Modeling Based on Polygonal Meshes," Association for Computing Machinery Special Interest Group on Graphics and Interactive Techniques, ACM SIGGRAPH, San Diego, 2007.

[2]Botsch, M., "High Quality Surface Generation and Efficient Multiresolution Editing Based on Triangle Meshes," Ph.D. Dissertation, RWTH Aachen, 2005.

[3]Hicks, R. M., and Henne, P.A., "Wing Design by Numerical Optimization," *Journal of Aircraft*, Vol. 15, 1978, pp. 407-412.

[4]LeGresley, P., and Alonso, J., "Investigation of Non-Linear Projection for POD Based Reduced Order Models for Aerodynamics," AIAA Paper 2001-0926, 2001.

[5]Samareh, J., "Aerodynamic Shape Optimization Based on Free-Form Deformation," AIAA Paper 2004-4630, 2004.

[6]Samareh, J., "Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization," *AIAA Journal*, Vol. 39, No. 5, 2001, pp. 877-884.

[7]Desideri, J.A., and Janka, A., "Hierarchical Parameterization for Multilevel Evolutionary Shape Optimization with Application to Aerodynamics," International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, EUROGEN, 2003.

[8]Jakobsson, S., and Amoignon, O., "Mesh Deformation using Radial Basis Functions for Gradient-based Aerodynamic Shape Optimization," Swedish Defense Research Agency Technical Report FOI-R–1784–SE, 2005.

[9]Fudge, D., Zingg, D., and Haimes, R., "A CAD-Free and CAD-Based Geometry Control System for Aerospace Shape Optimization," AIAA Paper 2005-0451, 2005.

[10]Persson, P.O., Aftosmis, M.J., and Haimes, R., "On the Use of Loop Subdivision Surfaces for Surrogate Geometry," 15th International Meshing Roundtable, 2006.

[11]Dube, J.F., Guibault, F., Vallet, M.G., and Trepanier, J.Y., "Turbine Blade Reconstruction and Optimization Using Subdivision Surfaces," AIAA Paper 2006-1327, 2006.

[12]Pressley, A., *Elementary Differential Geometry*, Springer Undergraduate Mathematics Series, 2001.

[13]Botsch, M., Pauly, M., Gross, M., and Kobbelt, L., "PriMo: Coupled Prisms for Intuitive Surface Modeling," Eurographics Symposium on Geometry Processing, 2006.

[14]Davis, T.A., and Hager, W.W., "Row modifications of a sparse Cholesky factorization," *SIAM Journal on Matrix Analysis and Applications*, Vol. 26, No. 3, 2005, pp. 621-639.

[15]Davis, T.A., and Hager, W.W., "Multiple-rank modifications of a sparse Cholesky factorization," *SIAM Journal on Matrix Analysis and Applications*, Vol. 22, No. 4, 2001, pp. 997-1013.

[16]Davis, T.A., and Hager, W.W., "Modifying a sparse Cholesky factorization," *SIAM Journal on Matrix Analysis and Applications*, Vol. 20, No. 3, 1999, pp. 606-627.

[17]Davis, T.A., and Hager, W.W., "Dynamic supernodes in sparse Cholesky update/downdate and triangular solves", *ACM Transactions on Mathematical Software*, Vol. 35, No. 4, 2008.

[18]Chen, Y., Davis, T.A., Hager, W.W., and Rajamanickam, S., "Algorithm 8xx: CHOLMOD, supernodal sparse Cholesky factorization and update downdate," *ACM Transactions on Mathematical Software*, Vol. 35, No. 3, 2008.

[19]Jiao, X. "Volume and Feature Preservation in Surface Mesh Optimization," 15th International Meshing Roundtable, 2006.

[20]Botsch, M., and Kobbelt, L., "A Remeshing Approach to Multiresolution Modeling," Eurographics Symposium on Geometry Processing, 2004.

[21]*The Stanford 3D Scanning Repository*, Stanford University Computer Graphics Laboratory, http://graphics.stanford.edu/data/3Dscanrep/, Date of Last Access: August 9, 2008.

[22]*MeshLab*. A tool developed with the support of the Epoch NOE, http://meshlab.sourceforge.net, Date of Last Access: August 10, 2008.